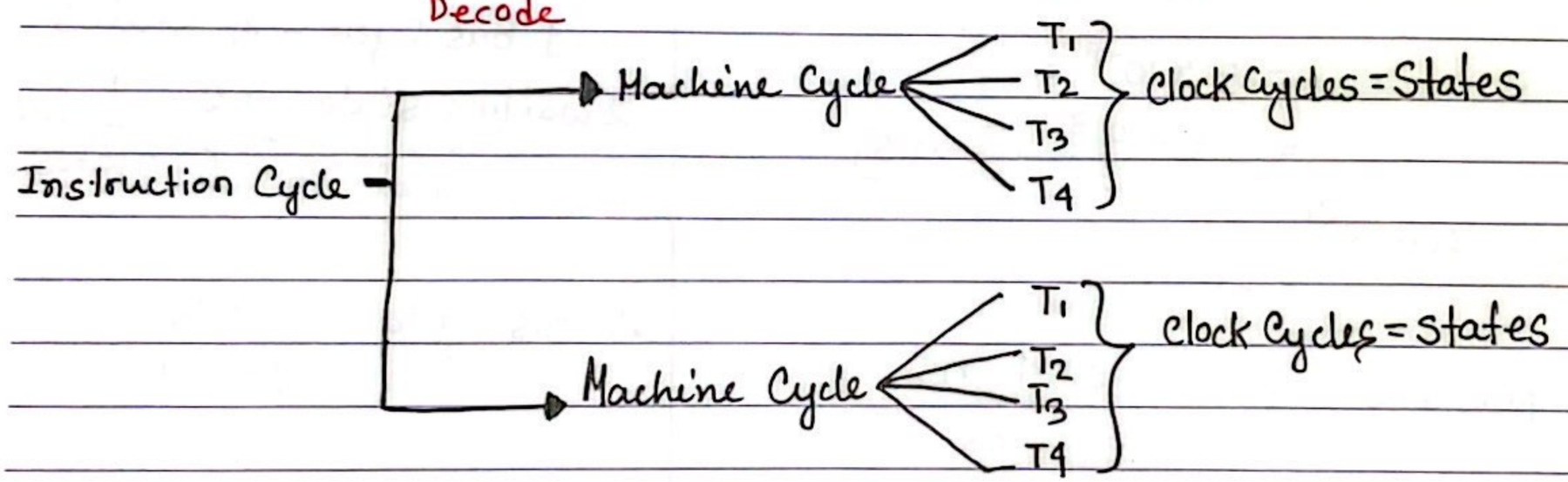
After Mid CSE341 (L1)

Fetch + Decode
Execute
} Total time = Instruction cycle = 1/None Bus/Machine cycle.

Fetch + Decode time = Bus/Machine cycle = At least 4 clock cycle (Each C.Cycle = 1 state) T_1, T_2, T_3, T_4

Execute time = Bus/Machine cycle = At least 4 clock cycle.

Instruction Cycle = Fetch cycle (FC) + Execute cycle (EC)
(Fetch + Decode)



Clock Generation: Clock for synchronization.

Clock generator circuit = 8284A | pin 19 (CLK) of 8086.

System Clock Concept

Time Should be calculated in NS (NenoSeconds)

Frequency: Cycle completed within a Second.

→ **Time Period**: $\frac{1}{\text{Frequency}}$ = Time for 1 clock state

1 **Example**: if frequency = 20 MHz

then time for 1 clock state = $\frac{1}{20\ \text{MHz} \times 10^6}$ [MHz → Hz]

$= 5 \times 10^{-8}$ s

$= \frac{5 \times 10^{-8}}{10^{-9}}$ [s → ns]

→ [1 Bus cycle/Machine Cycle = 4 c·State]

$= 50$ ns

∴ 1 Bus Cycle/Machine Cycle = 50×4 ns
= 200 ns

2 **Example** Mov Ax, [22465h] at 80MHz

a) Calculate one clock state duration?

b) " The duration of one bus cycle if there are two waiting states.

c) Calculate the duration of the Instruction cycle.

Solution Example 2

(a) Duration of one clock state =

$\frac{1}{f} = \frac{1}{80 \times 10^6}$ s

$= \frac{1.25 \times 10^{-8}}{10^{-9}}$ Ns

$= 12.5$ ns.

(b) There are two waiting states.

1 Bus Cycle = 4 states.

2 waiting state = +2 "

Total = 6 states.

∴ Bus cycle = 6 × 12.5 ns
= 75 ns

(c) Mov Ax, [22465h]

here only fetch, No execution as it is only MOV.

So,

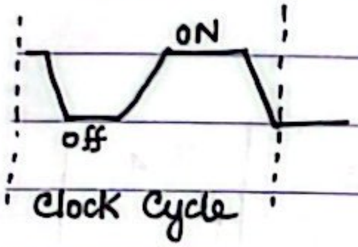
1 Bus cycle = Instruction Cycle.
= 75 ns.

We Know,

Ins cycle = 1/More Bus cycles.

Date: / /

→ Duty Cycle = $\frac{T_{ON}}{T_{ON} + T_{off}} \times 100$



Duty Cycle means for how long the clock cycle was ON.

→ By default Duty Cycle is 33%

T3 and T4 এ কতক্ষণ Data থাকবে তার উপর depend করে RD and DEN

→ Clock States :

(AD_0 - AD_{15}) Multiplex (Address + Data)

T_1 = AD_0 - AD_{15} (Address flow)

T_2 = Mem, I/O, read/write

T_3 = Data is supplied

T_4 = Control Signal Remove.

→ Read Bus Timing Diagram : MUST

20bit { A_{D0} - A_{15} → T_1 → Address flow (1/0)
A_{16} - A_{19} → (S3 - S6)

ALE → (Address Letch Enable)
parallel when Address available (High)

$M/\overline{IO}$ → M → 1 to 0
I/O → 0 to 1

$DT/\overline{R}$ → Bar means 1 start then 0

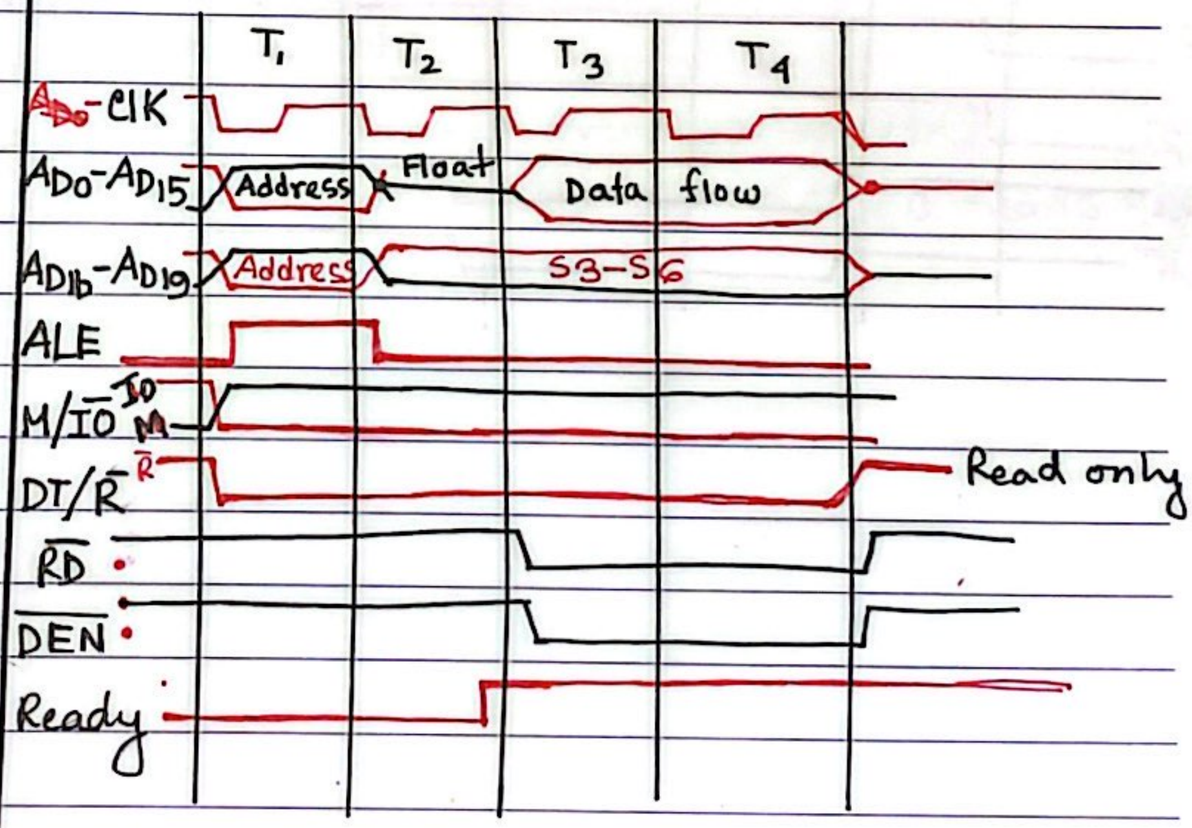
} Depends on the question R or Transmit, Mem/IO

Ready → when mem to CPU
Check at the end of T_2 in AD_0 - AD_{15} if there is Data flow then parallelly Active the Ready.

$\overline{RD}$ → Read Bar means (1 to 0) when there will be data flow at AD_0 - AD_{15}.

$\overline{DEN}$ → when AD_0 - AD_{15} will flow Data, the Data enable pin will be 1(ON) to (off) 0 till data flow.

Timing Diagram for a read cycle



Zarin Haider

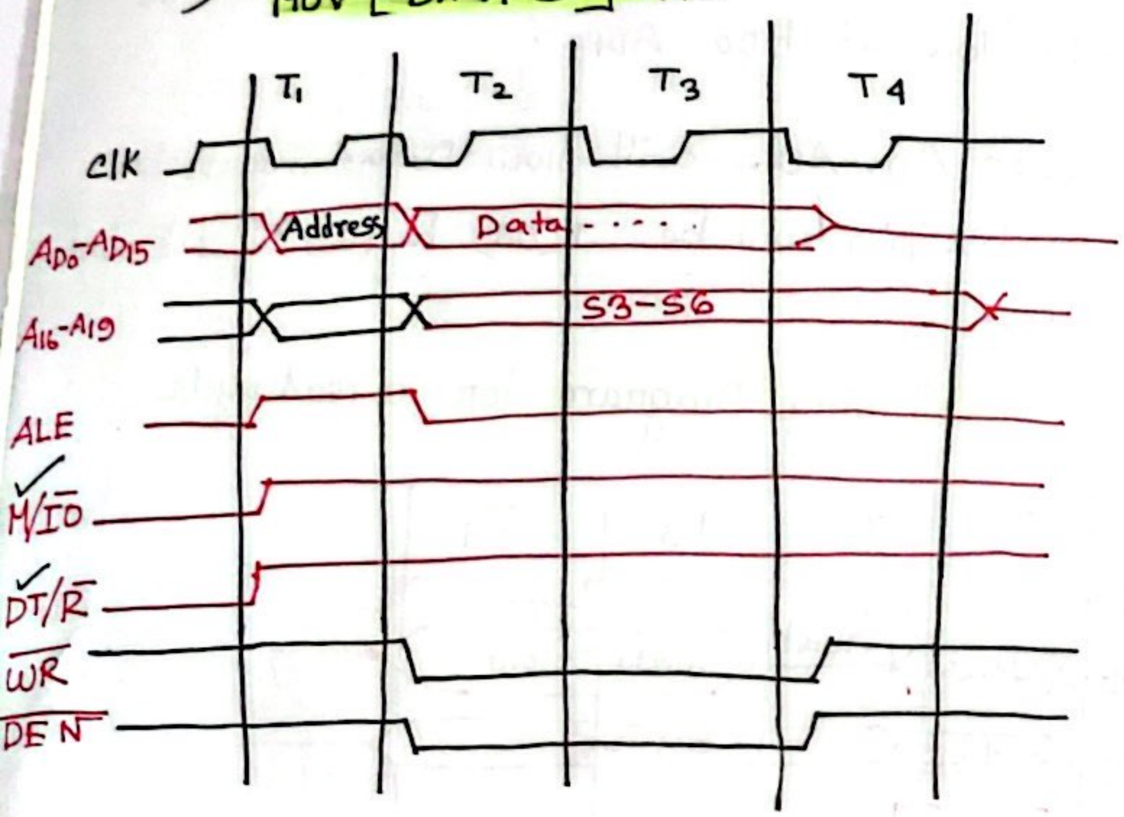
After Mid lecture-2

Write Cycle → No float in write.

→ There is no "ready" pin in write cycle.

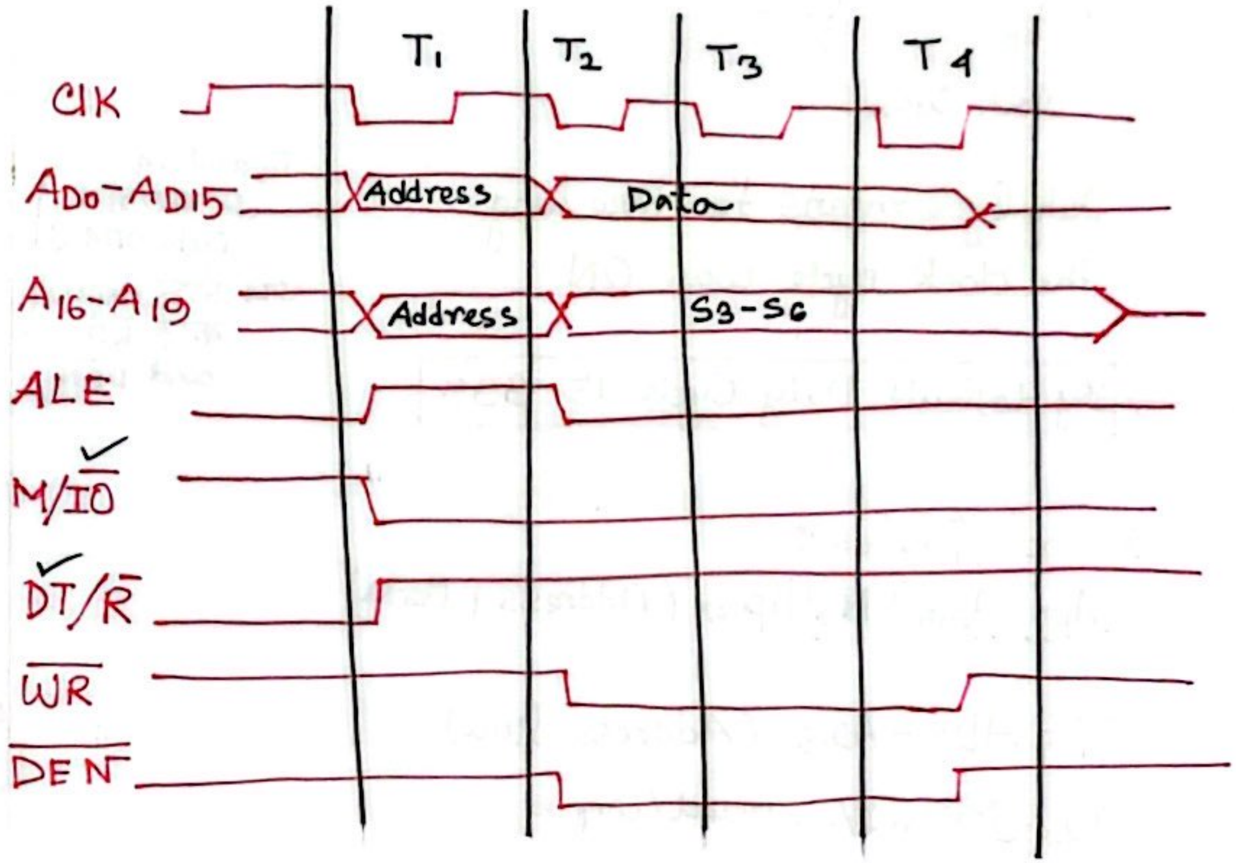
Write Cycle Example 1

i) Mov [Bx+SI], AX



Write Cycle Example 2

ii) Out Dx, Ax



Zarrinhaider

Date: / /

8086 Memory Bank

Address Bus → 20 Bit.

Data Bus → 16 bit.

1100 ..1100	→ each 8bit.
0110 1110	

8bit : 1 Byte

↳ Now Read or Write → it will take 1 Bus Cycle.

16Bit { 1Byte / 1Byte } 1 word → needs ~~to~~ 2 Bus cycles to R/W.

So, it needs (4X2) = 8 clock cycles.

→ 2 Bus Cycles means more time. So, try to take less time or 1 Bus cycle only.

So for taking less time we can divide the memory into two Banks. (Even/low Bank and odd/high Bank)

→ Consider the Address, Even or Odd.

Even Bank → All even addresses (5Hex)

Odd Bank → All odd addresses (5Hex)

Each Location will contain 8 bits as before.

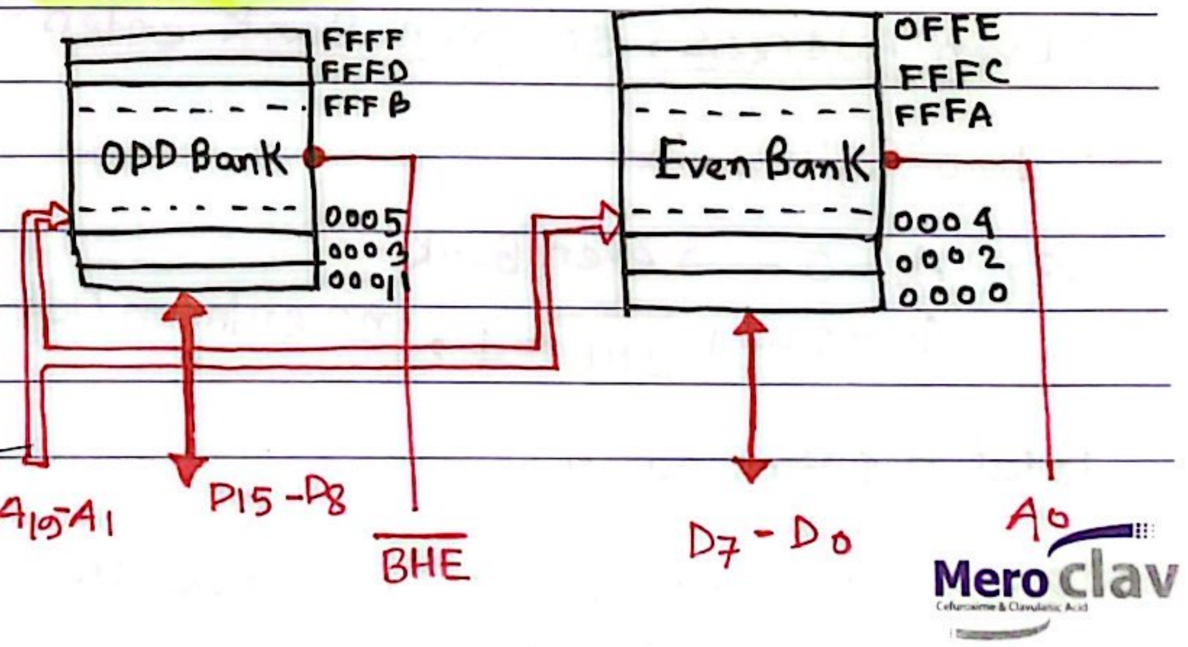
High Bytes		Low Bytes	
8 bit	FFFF	8Bit	FFFE
	FFFD		FFFC
	FFFB		FFFA
Odd Bank		Even Bank	
	0005		0004
	0003		0002
	0001		0000
D15 - D8 (high)		D7 - D0 (Low)	

Bank Control:

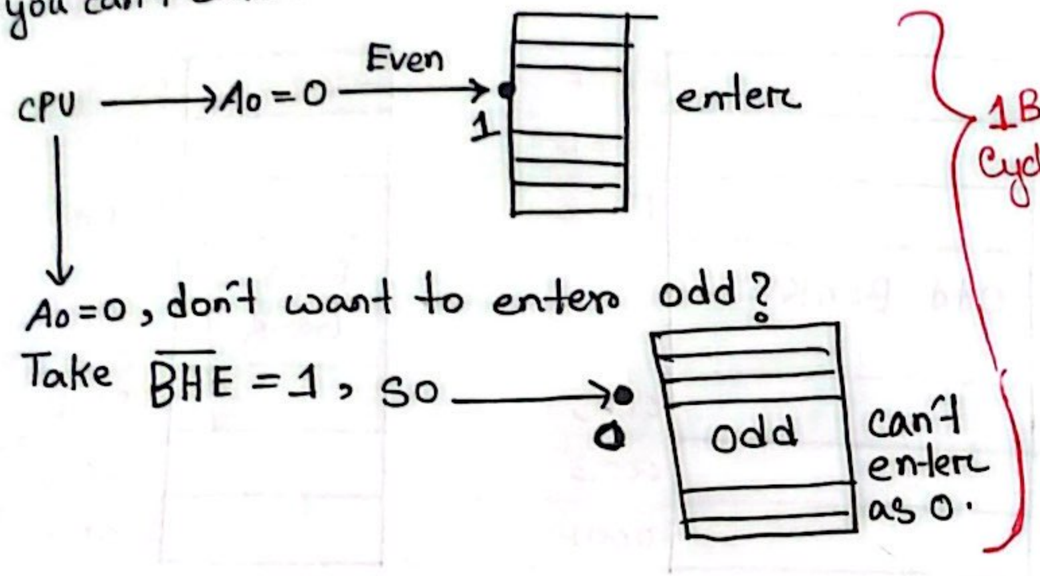
→ A0 Pin (Address Even/ODD indicate)

→ $\overline{BHE}$ → Bus Enable High pin (0 means high)

Must Draw



→ To enter any bank (Even/odd) you have to take A_0 and then the output should be 1. Otherwise you can't enter.



Example : Bank ✻ Mov AL, [24568h]

→ Even address as last digit 8.

→ will take 8 bit only, Because AL = 8bit Reg

→ As it is not 16 bit, so only one Bank.

→ Even Address, so even Bank enter.

→ Low Byte data.

So, $A_0 = 0 \longrightarrow$ even Bank

$A_0 = 0 \longrightarrow \overline{BHE} = 1$, no entry on Odd.

Total = 1 Bus cycly.

⇒ 8 bit = No profit or time save in 8 bits.

Example : Mov AL, [12333h]

$A_0 = 1$, so even → 0 (No entry)

$\overline{BHE} = 0$, so odd → 1 (entry)

} 1 Bus

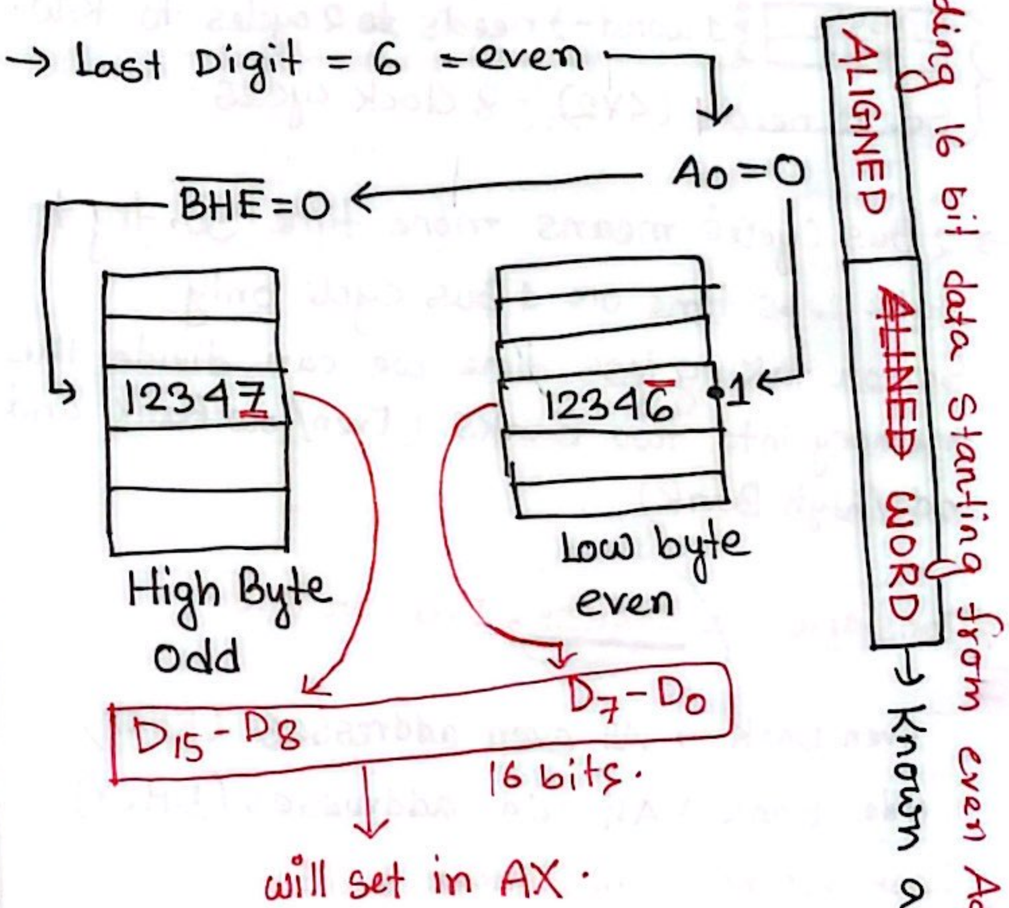
Example : Mov Ax, [2346h] (16bit)

→ 16 bit → even then odd → 1 bus cycle (Reading 16 bit data Starting from even Address)

or

odd then even → 2 bus ✓

→ Last Digit = 6 = even



1 Bus ✓

Date: / /

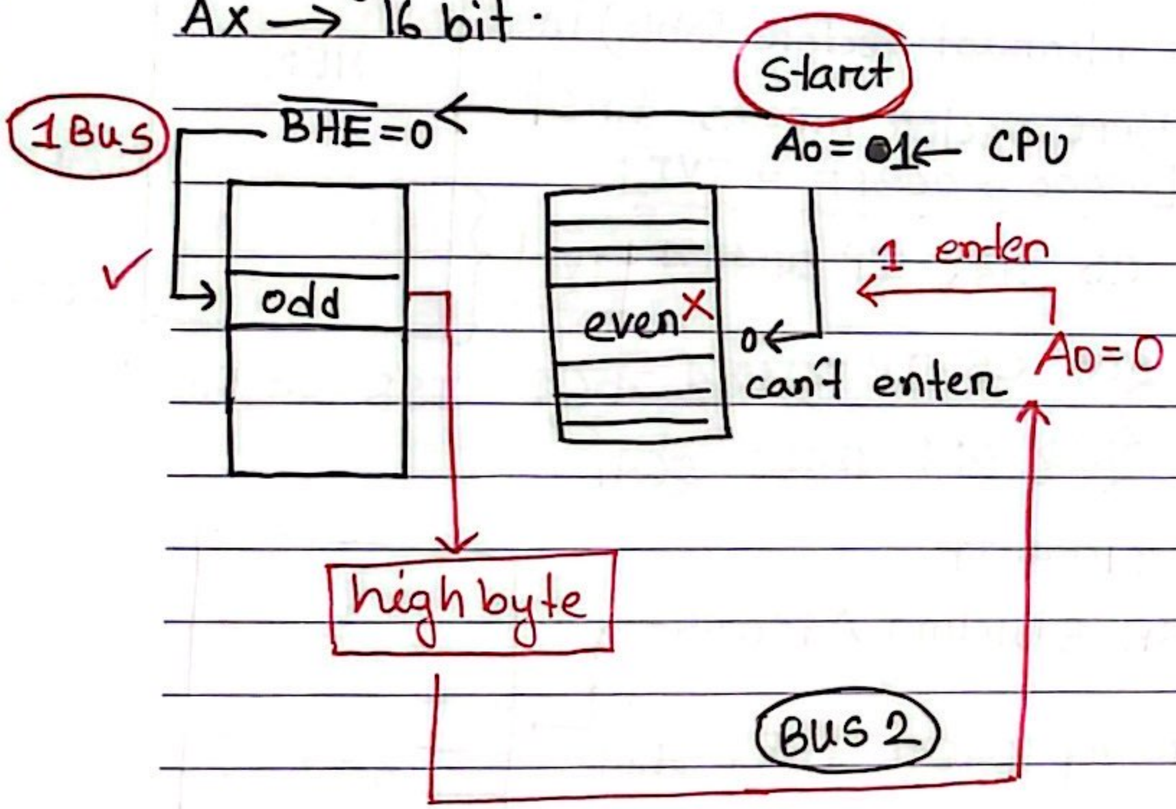
→ Reading 16 bit Data starting from odd bank takes 2 bus cycles. (can't be resolved)

→ *Nown as Unaligned word.

Example : Mov Ax,[103457h]

Last digit 7 → odd → start odd

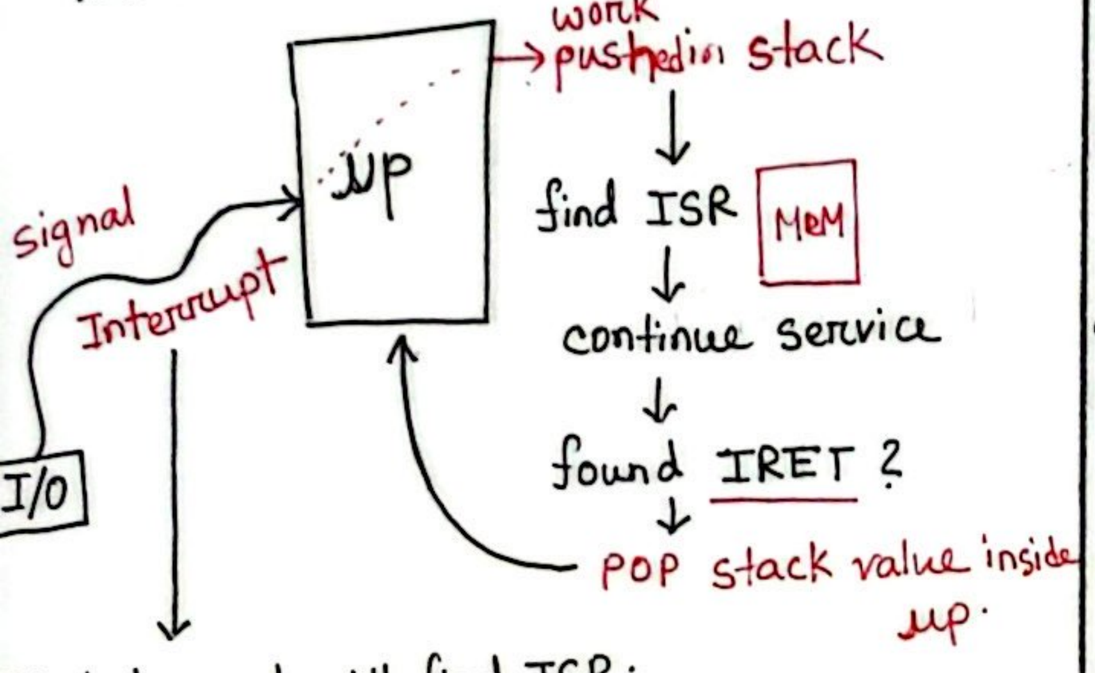
Ax → 16 bit.



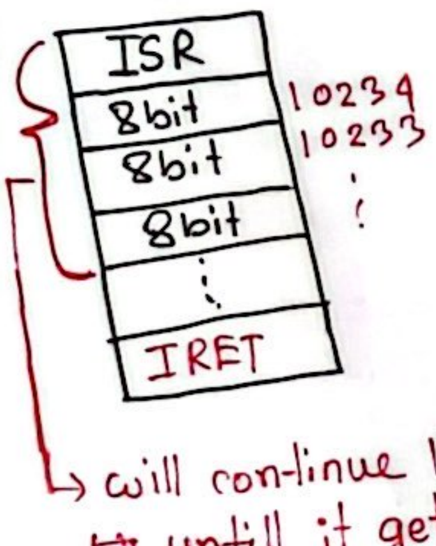
After Mid Lecture - 8 (Interrupts)

Suppose micro-processor is working.
Now a signal trying to enter

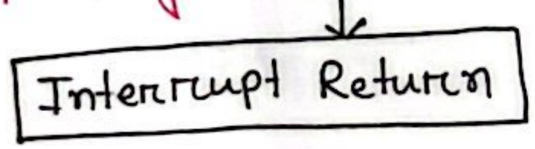
This is called interrupt.



This interrupt will find ISR.
ISR = Interrupt Service Routine.
↳ location → memory (RAM)



↳ will continue like this
untill it gets "I-RET"



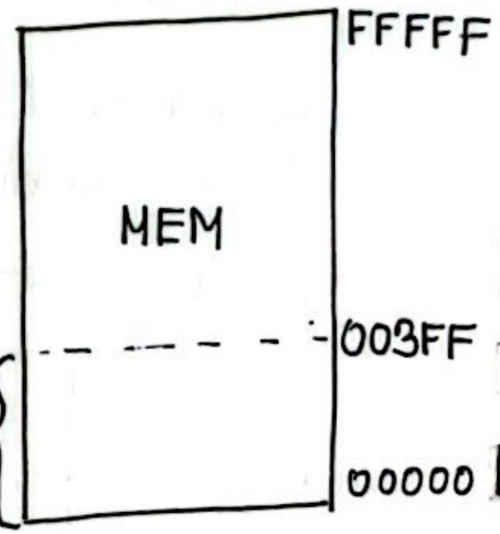
That means interrupt serving is done.

Interrupt কিভাবে কাজ করে?

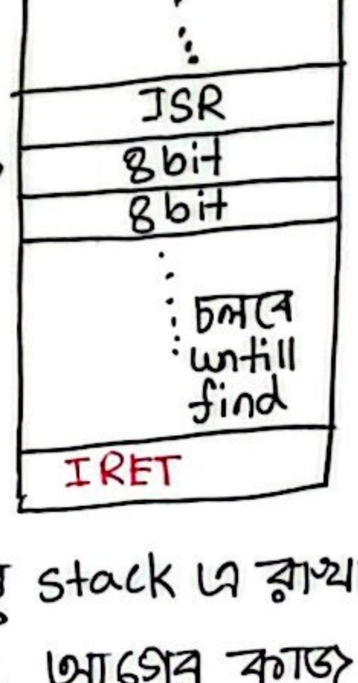
ধরুন µp কোনো কাজ করছে। তখন Anturrupt হলো। µp তখন Ongoing কাজকে push করে Stack এ রাখবে।
Suppose, "Int 60" (অনেক প্রকার হতে পারে Ant)
সেটি এখন ISR (Interrupt Service Routine) খুঁজবে Mem (RAM) এ।

ISR কিভাবে খুঁজে পাবে?

- Find IVT (Interrupt Vector Table) in Memory. • Here vector means first address. [00000 → 003FF = IVT]
- IVT তে CS এবং IP থাকবে IVT
- এই IVT হতে CS, IP ব্যবহার করে ISR এর First address খুঁজে বের করতে হবে।



P.A = CS x 10 + IP
= ISR Starting Address



- ISR starting থেকে IRET মানে Interrupt Return option পাওয়া পর্যন্ত service চলবে। IRET পেলে Service শেষ।
- I-RET হলে µp আবার push করা stack এ রাখা আগের কাজকে pop করবে এবং আগের কাজে continue করবে।

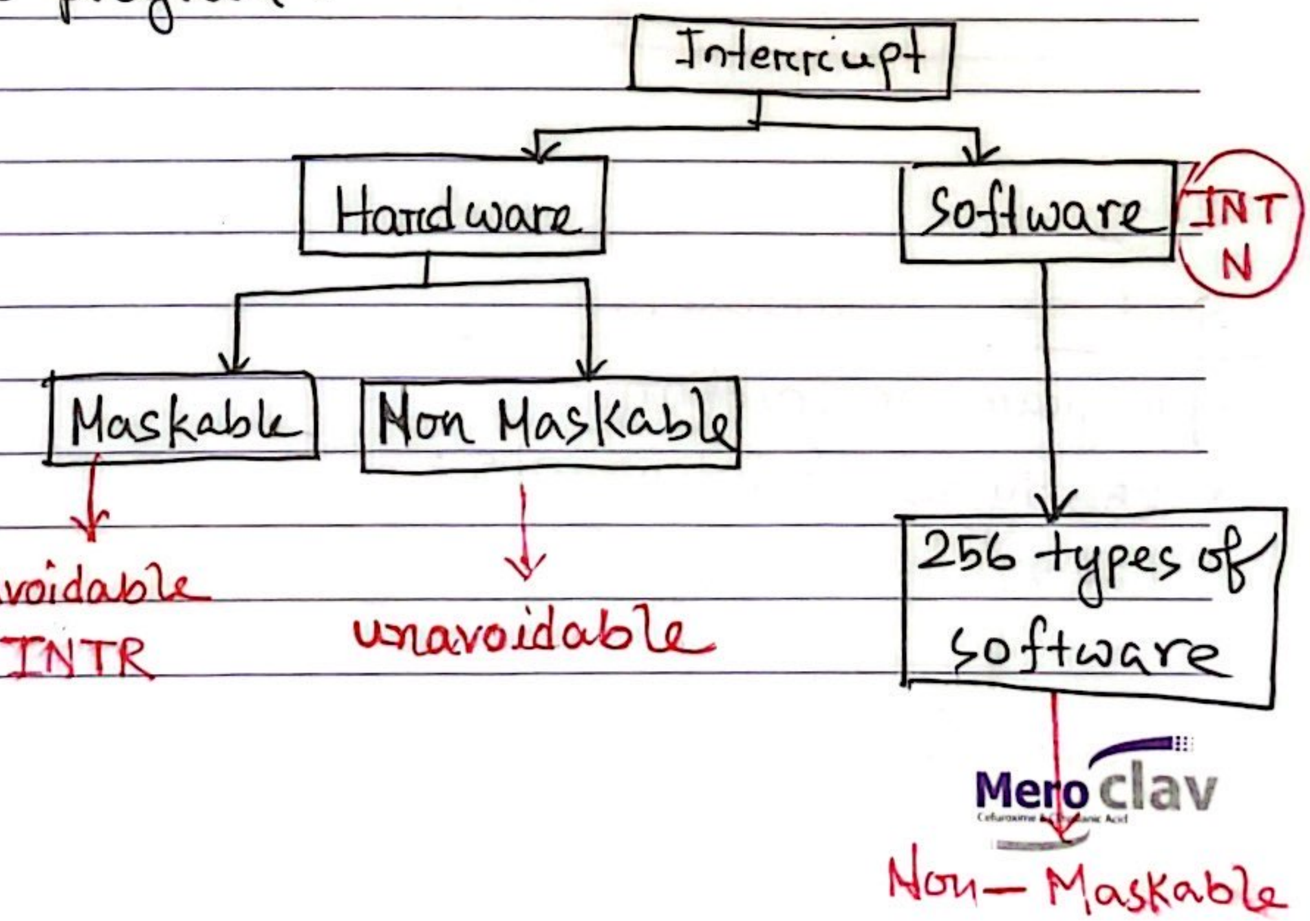
Interrupt

Date: / /

→ Interrupt the current work of cpu.
→ Basically it is an external signal/instruction.
→ Microprocessor ~~interrupt~~ doesn't value the current work when interrupt is here.

→ What happens?
- Jumps to ISR to respond the current interrupt.
- Each interrupt has it's own ISR.
- After dealing with interrupt it returns to its' previous work.

→ Interrupt from:
1) External signal from Keyboard, Mouse, Printer...
2) from special instruction in the program.
3) internal signal



Hardware Interrupt

Keyboard, mouse, DVD, hard disk .... etc.

- 8086 has two interrupt pins.

INTR (18)	NMI (17)
↓	↓
Maskable	Non Maskable

Maskable interrupt

- Served/wait/denied.
- INTR PIN
- Doesn't save stuffs.

Non Maskable

- All software interrupt.
- Serve immediately.
- NMI Pin
- Save things.
- Response in critical time

Software Interrupts

- program instruction.
- 256 types of interrupts.
  (00 → ff) or (0-255)

n

use as: Int n.

Classifications of 8086 interrupt

1. user defined available.
2. software " "
   - External signal.
   - Known as hardware int.
3. Pre defined Int.

Interrupt Vectors and Vector table: (IVT)

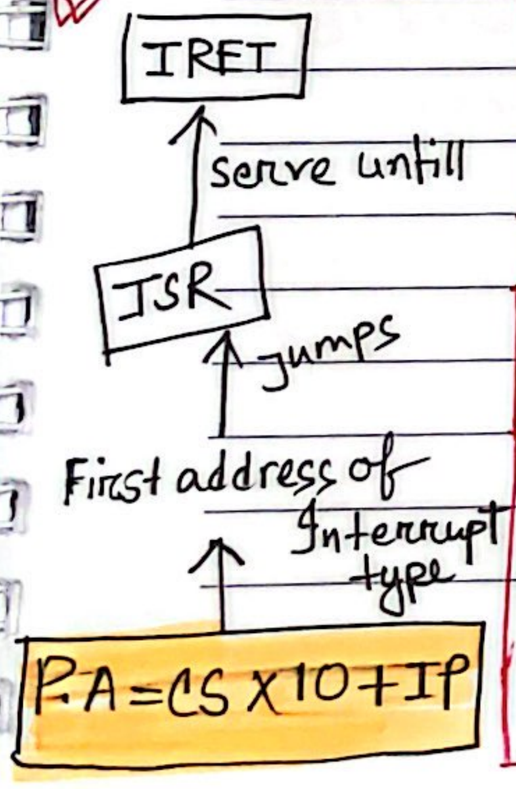
- Within 1 K.B of memory.
- (00000 to 003FF)

Interrupt Types based on ISR:

Date: / /

IVT → Vector table helps to find P.A / first address of ISR.

003FF H
00080 H
.
00014 H
00010 H
0000CH
00008 H
00007 → CS
00005
00004 H → IP
00003 → 48H → high byte → CS
00002 → 38H
00001 → 32H → low byte → IP
00000 → 20H



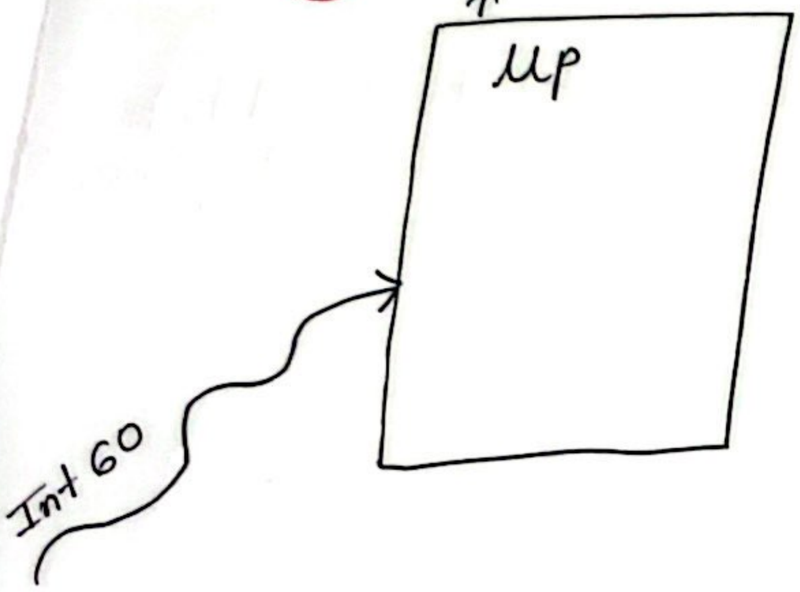
Type 0 (location 4)

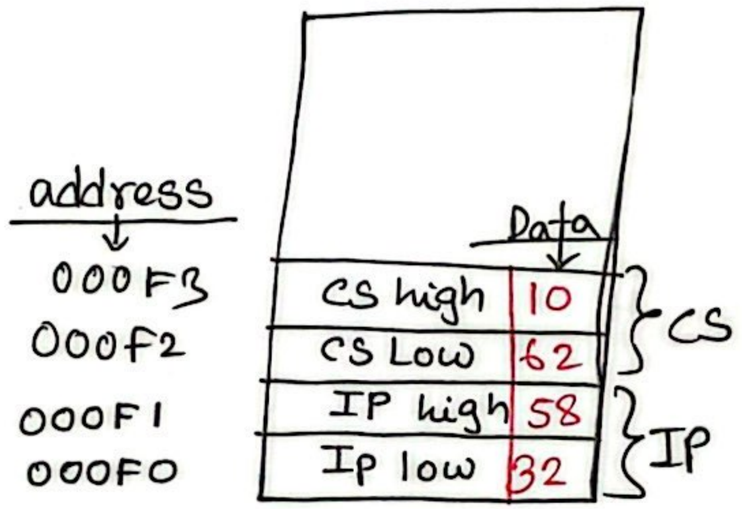
00001 00000
high low

So, IP = 3220 h

high low
4838
CS = (4838)

(1) Stack save current work





So, IP = 5832 h

CS = 1062 h

PA = CS x 10 + IP

= (1062 x 10) + 5832

= 16452 h.

= first address of ISR

↓ serve unit

Interrupt Return

Steps

① Int type x 4

= 60 x 4 = (240)d

= 00F0 (Hex)

= low byte address of IP

②

After Mid(L-4) Interrupt Part 2

Date: / /

→ Interrupt Vector = Interrupt Pointer.

→ (00000H to 000013 H) → Predefined (Type 0 → Type 4) ——→ total 5 type

→ (00014 H to 00070 H) ——→ Reserved (type 5 to type 31) ——→ total 27

→ (00080 H to 003FF H) ——→ Available for user (type 32-255) ——→ 224 total

VIP Explanation + Diagram

(How Interrupt is Served)

1) IntR entry. Interrupt করার পর trap flag (TP) resets.

সব 16 bit করে PSW, IP, CS

PSW → basically flag, current program saves in stack

segment → point offset

SS:SP

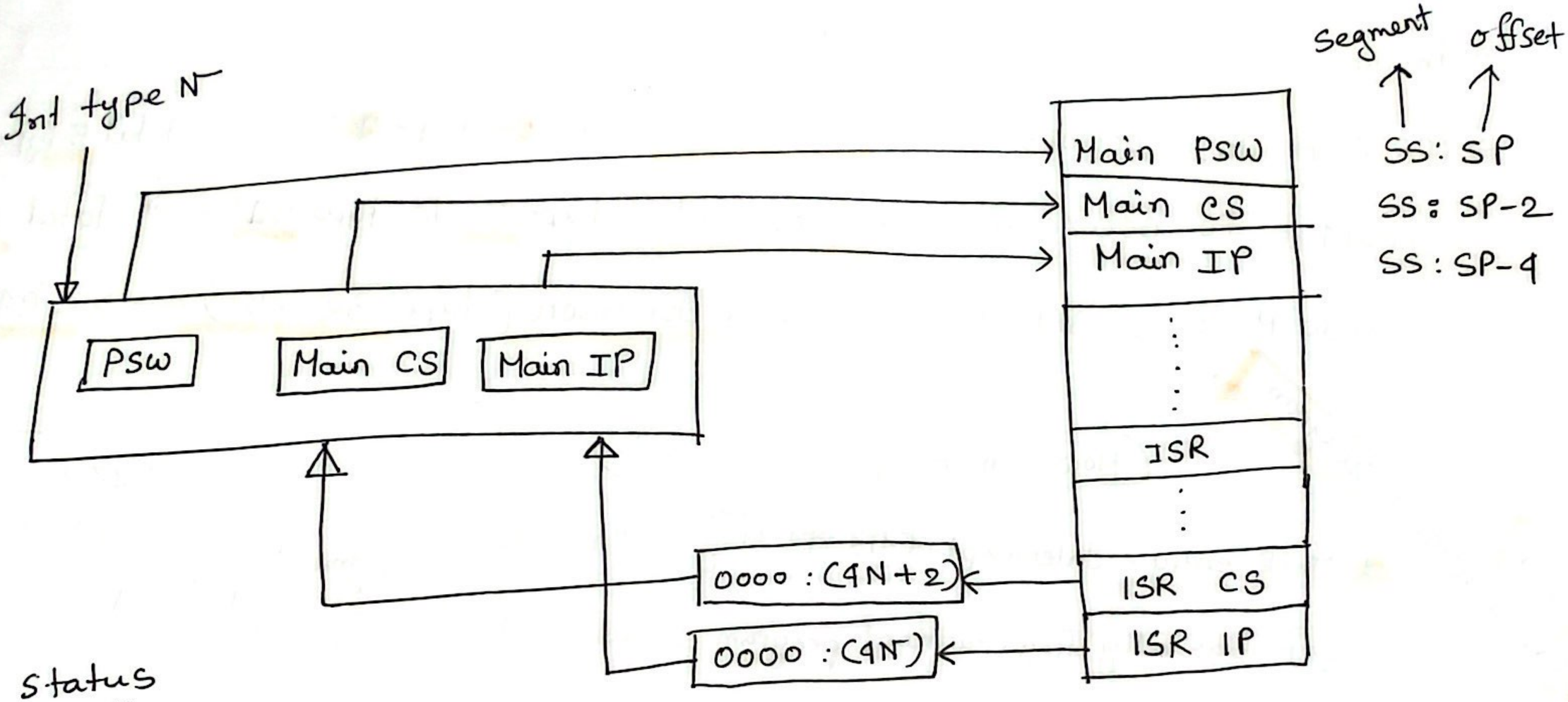
→ Current (Main) program flag use করে stack save করবে। তার জন্য stack এর segment এবং ratio point লাগবে। (SS: SP) → Main PSW

→ Flag এর পর দুই ঘর নেমে (SS : SP-2) CS কে Push করবে। → 8+8 = 16 bit

→ এরপর আবার দুই ঘর নেমে (SS : SP-4) IP কে Push করে।

Now, INT type x 4 = value in hexa = first address of ISR ⟹ then serve until IRET

Calculate P.A = CS x 10 + IP



Status while Executing main programmed

Minus(-2) → each box
8bit → 8x2=16

16 bit = PSW = CS = IP .

Date: / /

(If interrupt)

start

Main Program

1) Push Flags
2) Clear IF (Int flag)
3) Clear TF (trap flag)
4) Push CS
5) Push IP
6) Fetch ISR

→ works parallely.

start when Interrupt

Interrupt Service Routine (ISR)

Push registers

.
.
.

Pop registers

IRET

POP IP
POP CS
POP FLAGS

Executing Normal Work

Date: / /

Importance of Interrupt.

→ Before Interrupt MP used to ask each and every i/o devices about service.
[Polled I/O or Programmed I/O]

→ But now I/O devices askes the up if they need help.

Polling (when up used to ask others)

→ Simple, but slow
→ Need to check I/O up regularly.
→ Handled by CPU

Interrupt

→ fast but more complicated
→ processor is notified by I/O devices.
→ Handled by interrupt handler.
→

Self Study Slide 23-27

(23) Zero interrupt

→ Occurs auto when DIV/IDiv too large.

(24) Single Step Interrupt Type 1

→ stop after each instruction, waits for further direction

→ TF → type 1 (stack)

→ Disabled while doing interrupt service procedure.

→ (00004 to 00006h)

(25) Non maskable (Type 2)

→ when low to high transition (NMI)

→ starting → CS value.

→ Can't be disabled.

→ useful to save program (in case of power failure)

(26) Breakpoint Interrupt Type-3

→ implement a breakpoint function.

→ Start CS value — 0000EH

IP value start → 0000C H

→ Useful: debugging large programs.

when single stepping is inefficient

(27) Overflow (type 4)

→ if signed result is on two signed numbers. and too large to be represented.

CS start → 00012 H

IP " = 00010 H ✓

→ useful: detect overflow

→ two ways: (1) put jump to OF inst (JO) (2) Put int on of (INTO)

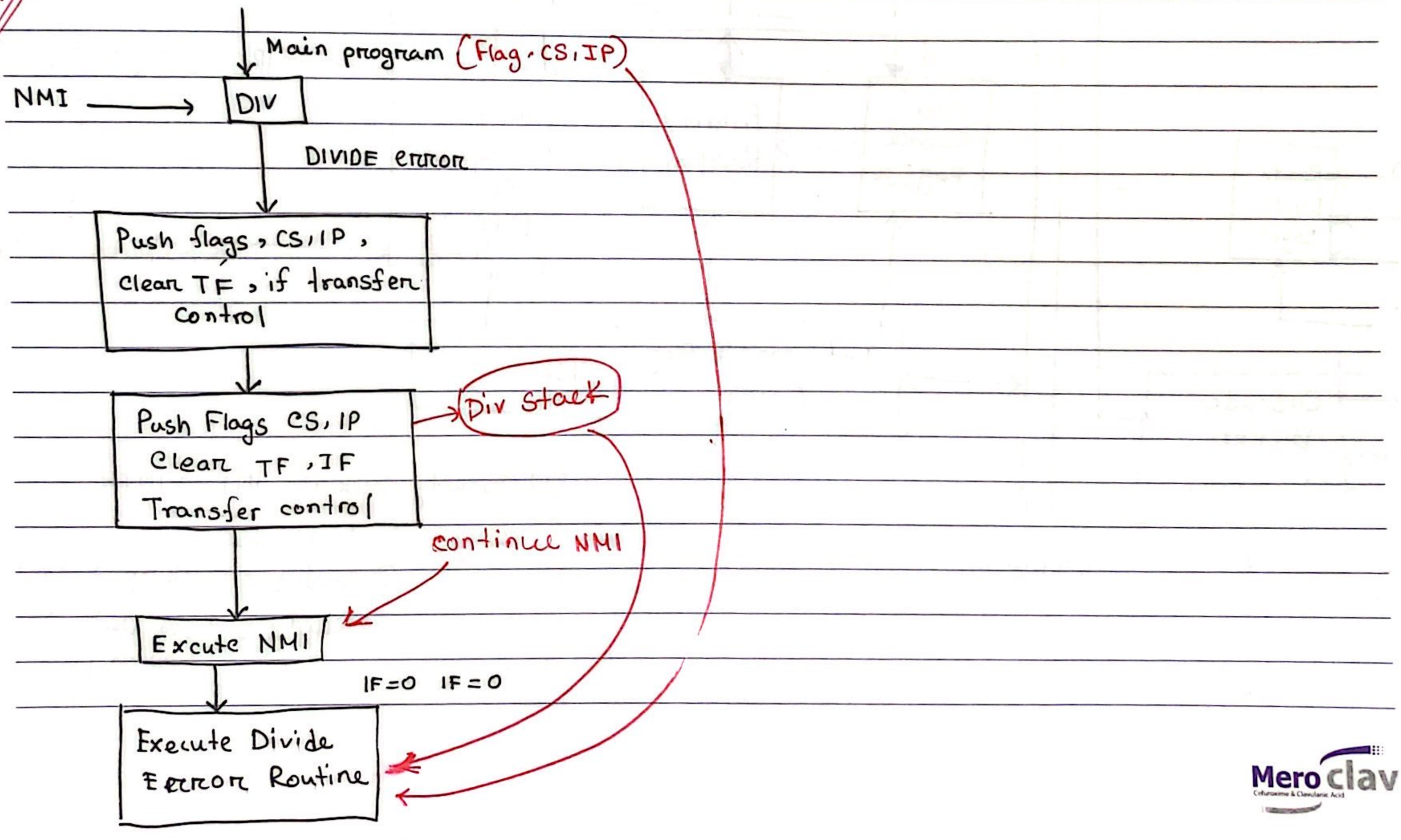
Date: / /

Highest Priority → DIVIDE ERROR > Intn > INTO > NMI > INTR → single step → Lowest Priority

If Divide and NMI at a time?

Ans: Divide error will be in stack.
then NMI. then again Divide.

IMPORTANT



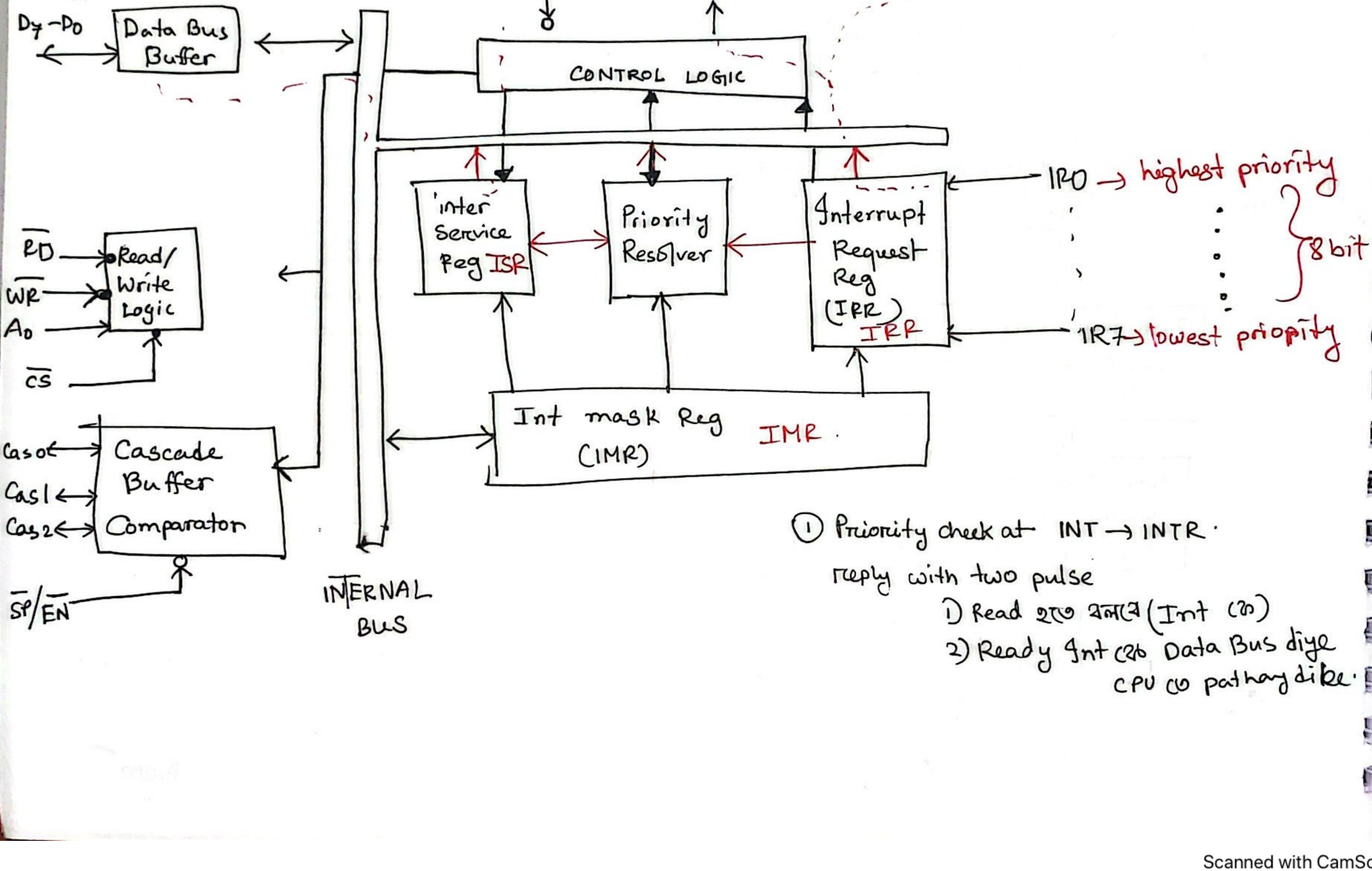
Programmable Interrupt Controller (PIC) 8259A

PIC → track রাখে

8086 Reset INTR

P.A IP.CS রাখে

8086



① Priority check at INT → INTR.

reply with two pulse

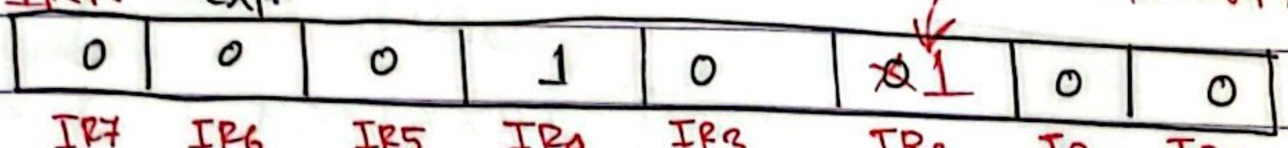
1) Read হলে বলবে (Int কর)

2) Ready হলে Int করে Data Bus দিয়ে CPU কে pathay dibe.

Registers

Date: / /

IRR Next



→ If IR4 has entry

→ check higher priority আছে নাকি

ISR

→ higher priority আসলে Signal ISR low priority কে বাদ দিবে।

IMR

→ IMR দিলে ISR হবে না। only IRR।

IMR → By default = 0

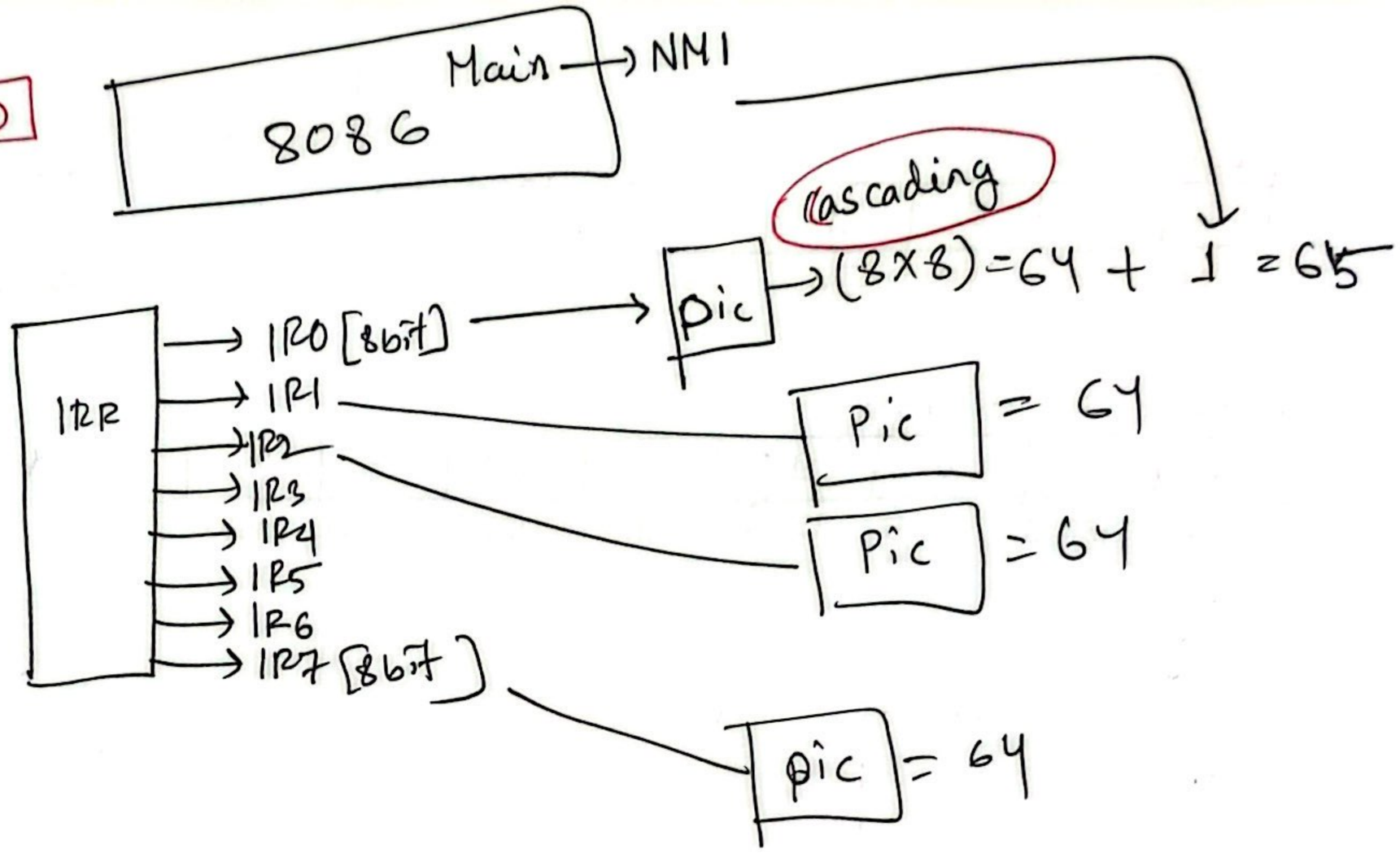
IMR যদি 1 হয় তাহলে ISR 1 হবে না, IRR হবে 1.

IRR value → ISR priority check → if bigger then 1 in ISR.

New IRR value → check high priority → if yes, then close last ISR, Set New ISR.

But if IRR less priprity then prev values wont change.

Cascading with 8259



So it can take more int at a time.
64 bit interrupt in each.

CAS 2	CAS 1	CAS 0	PIC
0	0	0	PIC1
0	0	1	PIC2
0	1	0	PIC3
0	1	1	PIC4
1	0	0	PIC5
1	0	1	PIC6
1	1	1	PIC7

I/O instructions

Date: / /

in → input
out → output

Two methods to define: → A0-A7

Fixed Address: 8Bit

byte known as P8.

Fixed because stored in ROM with OP code.

Example IN AX, 25 (here 25 fixed)

Variable Address: → A0-A15

it can be 8 bit / 16 bits, Register DX holds.

Because this can be changed.

Example: IN AX, DX

Here Dx means any value that will be stored.

So can't fix this.

I/O Address MAPPING (Accessing I/O Process1)

Advantage: Don't need to use I/O devices

Disadvantage: Memory space waste.

(এটি Basically memn এর একটি Location I/O এর জন্য ব্যবহার করা হয়। মানে O/I এর যেকোনো info এই Location এ রাখলে I/O device তা directly access করতে পারবে। এটা একটি common address I/O এবং memory এর জন্য।)

* Most of the processors use I/O Mapping.

I/O Mapping Advantage:

1. Less complications
2. Less circuitry
3. Same int such as Move can be used in Memory and I/O.
4. Less decoding.

Isolated I/O : (Process 2)

1) Separate I/O address থাকবে।
So Memory and I/O এর জন্য addressing fully আলাদা।

2) Separately MOV and In/out use.

3) IORC → I/O read cycle
IOWC → I/O write cyle
> I/O control এবং signal এর জন্য।

4) MEMR → for memory read
MEMW → for mem write.

5) Most common form in ITEL processor

6) Pentium supports isolated O/I of 64 KB address.

Isolated I/O Advantage:

1) No memory space wastage.

Disadvantage:

1) Separate signals.
2) আলাদা IN বা out instruction লাগবে।

3 different ways to transfer Data (I/O)

1) Programmed I/O
2) Interrupt-driven I/O
3) Direct memory access.

Programmed I/O :

এখানে CPU all the time I/O থেকে data পাওয়ার অপেক্ষায় থাকে অন্য কাজ বাদ দিয়ে। So CPU time waste করছে।

Adadvantage: Always in waiting until it gets I/O data. So time waste. Checking again and again for data from I/O. Here CPU performance degrades.

Interrupt-driven I/O :

it is better than programmed I/O

working process: CPU নিজের কাজ নিজের মতো করতে থাকে। যখনই I/O থেকে data আসে তখন Interrupt হয় এবং CPU আগে কাজ stack এ রেখে interrupt কাজ করে।

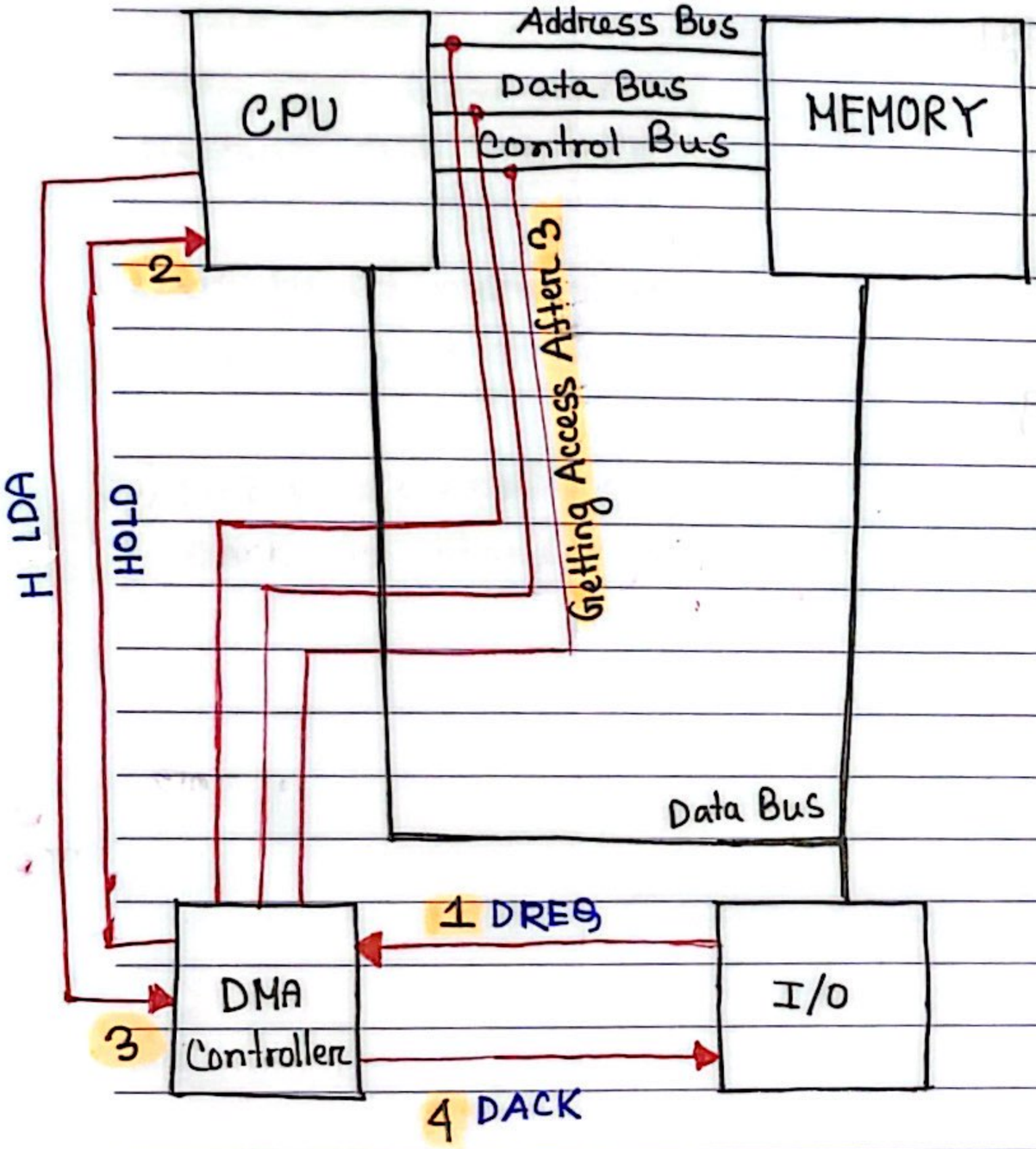
* Programmed I/O and Interrupt-drive এ CPU নিজে directly কাজ করে। As a result CPU এর কাজের pressure বেড়ে যায় এবং CPU এর অন্যান্য অনেক কাজ থাকে। তাই CPU এই কাজের জন্য assistant রাখে যাতে CPU কে help করতে পারে। which is known as **Direct Memory Access (DMA)**

Direct Memory Access (DMA)

H LDA

Important DMA OPERATION DIAGRAM

Date: / /



1 I/O প্রথমে DMA কে request করবে through DREQ (Direct Request)

2 then DMA will ask permission from CPU. [CPU যতক্ষণ না Request accept করবে ততক্ষণ "HOLD" এ রাখবে।
"HLQ" → Hold Request.

3 CPU request accept করলে তখন DMA কে Notification যাবে।
HLDA → Hold Acknowledgement.
তখন DMA directly (CPU to Mem) connection Bus/address/control Bus পেয়ে যাবে।

4 এখন DMA থেকে I/O কে "DACK" Notification যাবে। DACK → DMA acknowledgment.

DMA Controller

→ 4 channel device (মানে ৪টা DREQ option থাকবে। Each DREQ is dedicated to a specific I/O Device.
→ Example: Intel 8237. → can address 64 KB. → Can decide priority of request.

Total 16 bit ⟶

DB7	DB0
8 bit	

→ upper level of Lower byte.

$A_7 - A_4$	$A_3 - A_0$
8 Bit	

Upper level nibble — Lower level nibble

8237 DMA Controller

Pin DREQ 0 → DREQ 3 খুবই Request দেয়ার কাজ করে।

DACK 3 → DACK0 Acknowledges DMA request.

HRQ (hold Request) Request control.

HLDA (Hold Acknowledge) Acknowledge hold Req;

AEN (Address enable)

DHA Cascade Mode

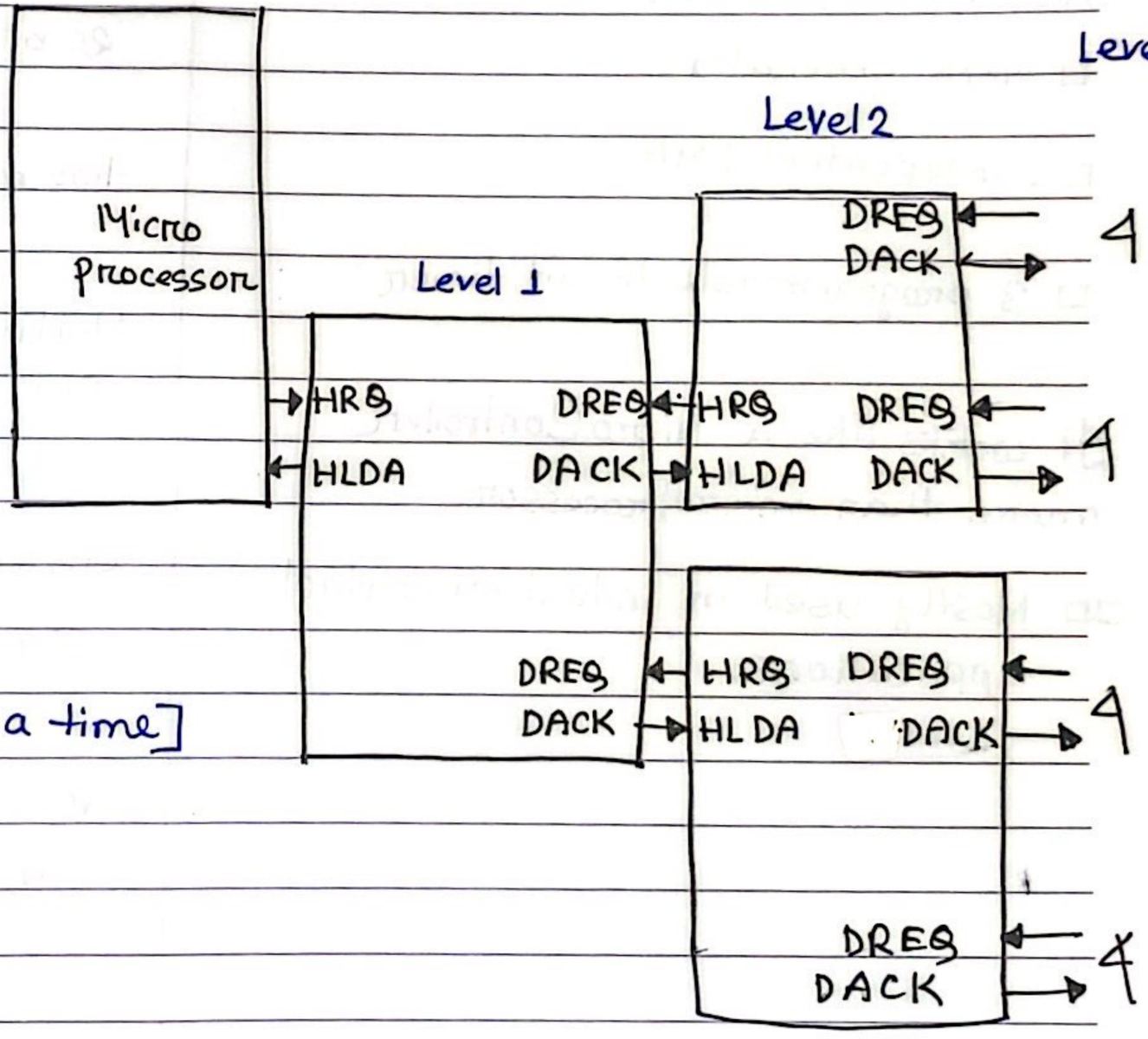
Date: / /

1) এটার মাধ্যমে at a time
অনেক গুলো 8237 add করে
এক সাথে কাজ করা যায়।

2) Utility of more than
4 channels at a time.

3) DREQ is connected with HRQ
DACK is connected with HLDA.

[16 I/O Devices at a time]



(ঢ) 16

Last Topic 80186

→ Data Bus → 16 bit.
→ Address Bus → 20 bit.

Additional functions:

- clock generators
- 2 independent DMA
- 3 programmable 16 bit timer.

It works like a MicroController more than microProcessor.

- Mostly used in industrial control applications.

8086	80286
16 bit HP	16 bit HP
16 bit Data Bus	16 bit data Bus
20 bit address Bus (1MB)	24 bit address bus (A_0-A_{23}) (16MB)
two modes (Max/Min)	2 Modes (Real/virtual)
Multiplexed Bus. So pin 40	Non Multiplexed bus So 68 pins.
Slower: 5 to 10 MHz Standard rate 6 MHz	Faster: (10-12.5) MHz Standard rate 8MHz
No memory management and protection	High performance with memory management and protection.

Real: এই MODE এ 80286 সবকাজ 8086 এর মতো। কিন্তু Faster. 6times
Extra 4 bit (24bit থেকে) ব্যবহার করতে পারে না।
protection and Management নেই।

Protected Virtual Mode: সব করতে পারে।